Overview of Core Flight System (CFS) Implementation of the Goddard Mission Services Evolution Center (GMSEC) Reference Architecture

Jonathan Wilmot
GSFC Code 582
Jonathan.j.wilmot@nasa.gov

What is CFS?

The Core Flight System is a platform-independent, reusable Flight Software (FSW) environment integrating a core flight executive, software component library, and Integrated Development Environment (IDE). Developed using key concepts of the GMSEC reference architecture

- **Layered Architecture**
- **Standard Middleware/Bus**
- **Application Programmer Interface**
- **Plug and Play**
- **Reusable Components**

} Core Flight Executive

} Component Library

Why is CFS Important?

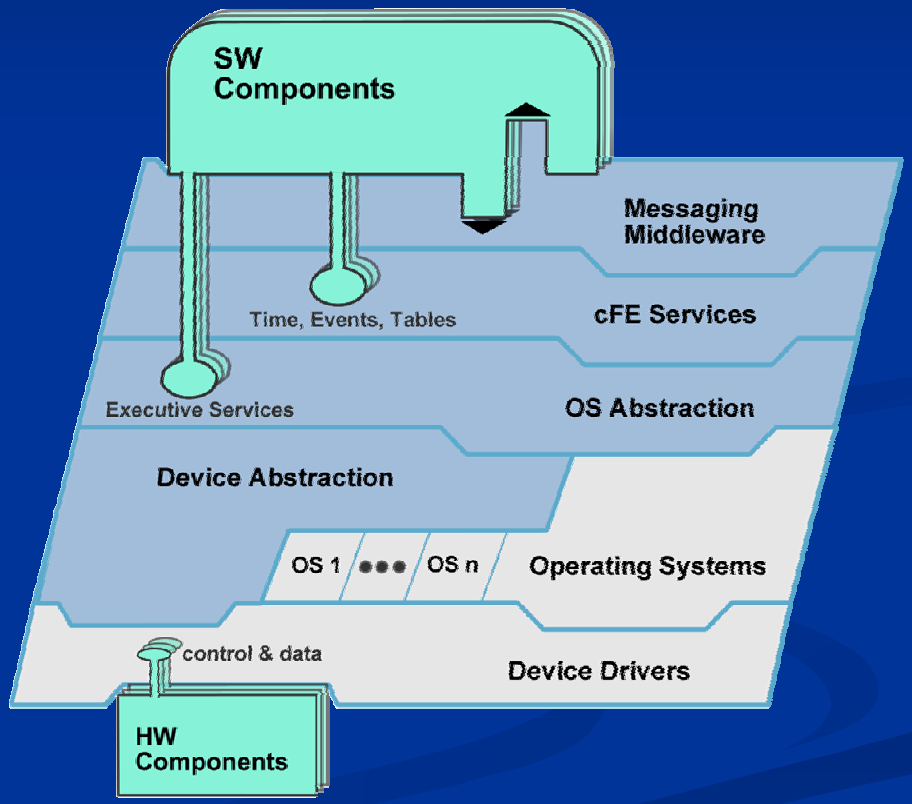
- CFS is a product based on an engineering strategy with the capability to:
  - Reduce Time to Flight
  - Reduce Risk
  - Increase Flight Application Capabilities
  - **Directly facilitate formalized software reuse**
  - **Enable collaboration across organizations**

Key Concept #1

Scalable Layered Architecture Isolates Dependencies

- **Layered Architecture**
- CFS internals are carefully layered.
- Each layer "hides" its implementation and technology details.
- Internals of a layer can be changed -- without affecting other layers' internals and components.
- Small-footprint, light-weight architecture and implementation minimizes overhead.

- **Benefits:**
- Enables technology infusion and evolution.
- Doesn't dictate a product or vendor.
- Enables modification at all stages of development and on-orbit.
- Provides Middleware, OS and HW platform-independence.



Key Concept #2
Standard Middleware Bus

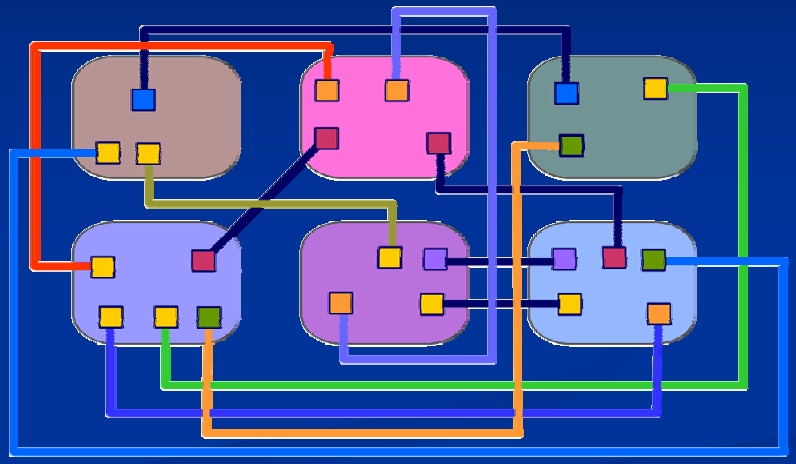
Publish/Subscribe

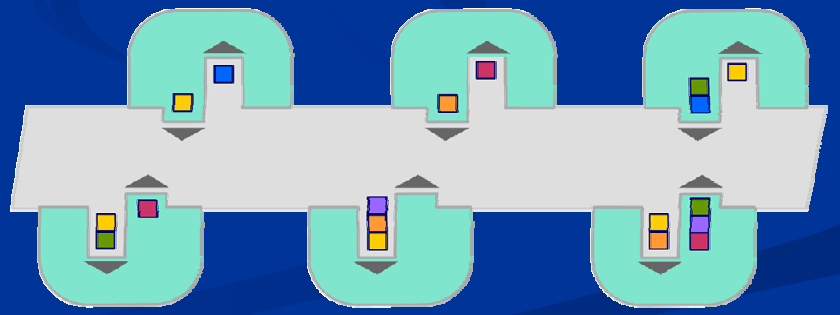
- **Components communicate over a standards-based Message-oriented Middleware/Software Bus**.
- The Middleware/ Software Bus uses a Publish/Subscribe model, so **cooperating components don't need to know the details of inter-communication** (location of others, protocols used, etc.).

Benefits:

- Simplifies component SW
- Minimizes interdependencies
- Supports HW and SW runtime "plug and play"
- Speeds development and integration.
- Enables dynamic component distribution and interconnection.

Legacy: Tightly-coupled, custom interfaces- data formats - protocols, internal knowledge, component interdependence





Publish/Subscribe: loosely-coupled, standard interface, data formats, protocols, & component independence

Key Concept #3

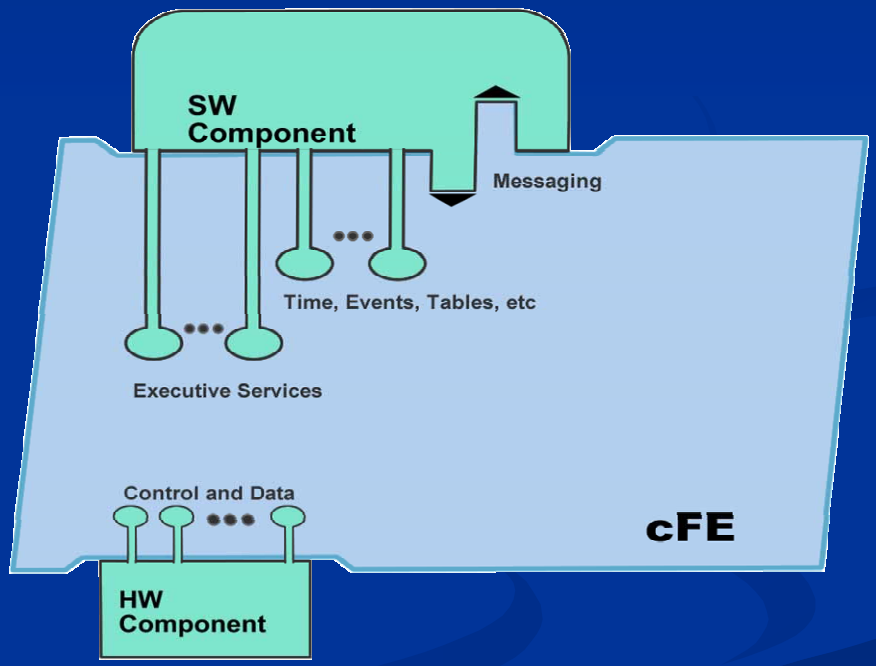
Standardized API for Software and Hardware Components

Application Programmer Interfaces

- CFS services and middleware communication bus has a **standardized, well-documented API**
- An **abstracted HW component API enables standardized interaction** between SW and HW components.

Benefits:

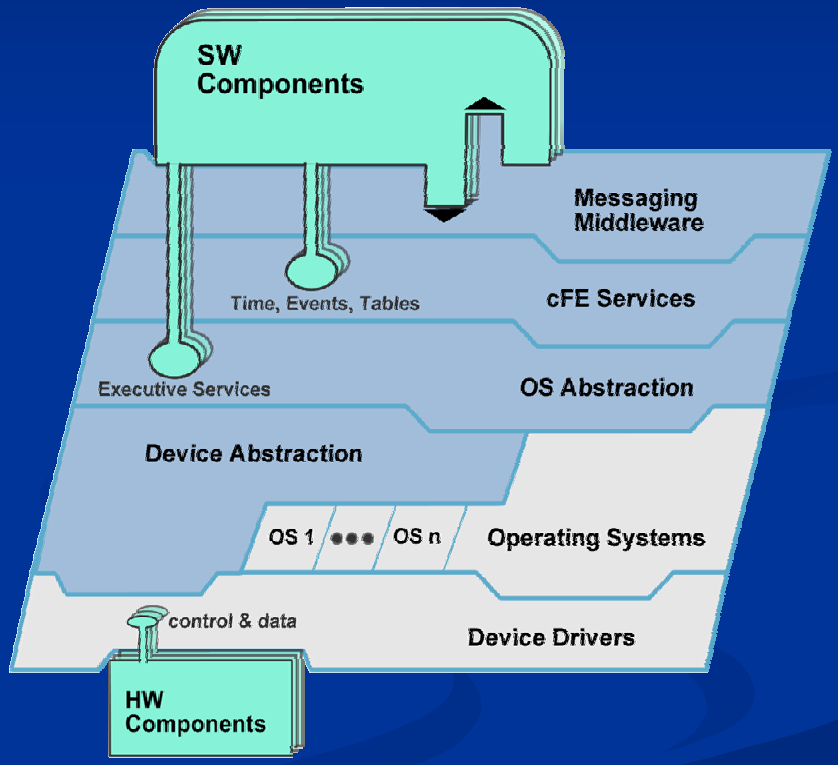
- Allows development and testing using **distributed teams**
- With the framework already in place, a**pplications can be started earlier** in the development process
- **Don't need to wait for the C&DH box** to be completed.
- **Simplifies integration, reduces development time, shortens schedules**.



API supplies all functions and data components developers need.

Packaging the first 3 Concepts
The
Core Flight Software Executive (cFE)

- Strategic Software Layering
  - Software of a layer can be changed without affecting the software of other layers
- Advanced Message Handling
  - Eliminates manual configuration of FSW
  - Automates integration of FSW with applications and hardware components (Publish/Subscribe model
- Standardized, Abstracted Interfaces
  - Minimizes software impacts from flight hardware, RTOS(*), and application changes



CFS Key Concept #4

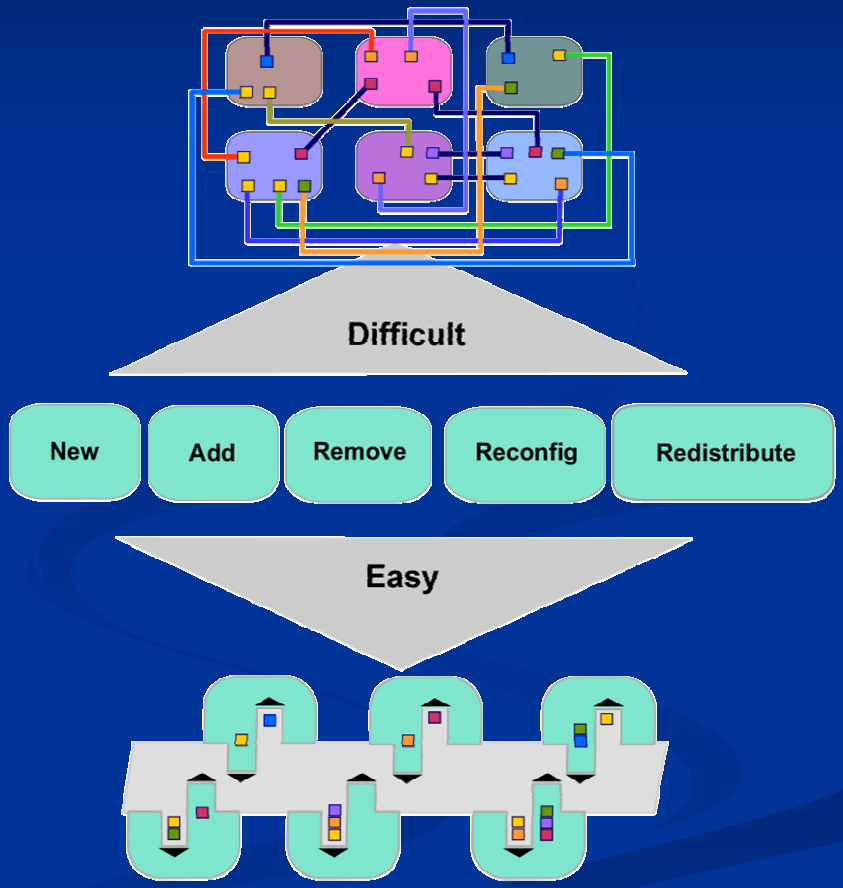
SW Components and HW Devices Plug and Play

Plug and Play

- cFE API's support add and remove functions
- SW **components can be switched in and out at runtime**, without rebooting or rebuilding the system SW.
- **Qualified Hardware and CFS-compatible software both "plug and play."**

Benefits:

- **Changes can be made dynamically** during development, test and on-orbit even as part of contingency management
- **Technology evolution/change** can be taken advantage of later in the development cycle.
- T**esting flexibility**

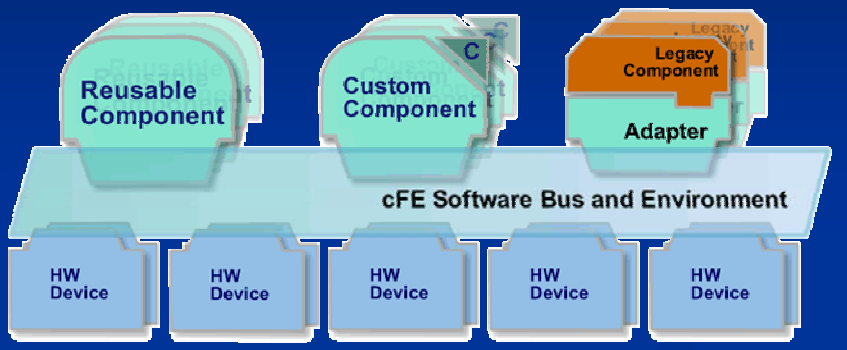


This powerful paradigm allows SW components to be switched in and out at runtime, without rebooting or rebuilding the system SW.

Reusable Components

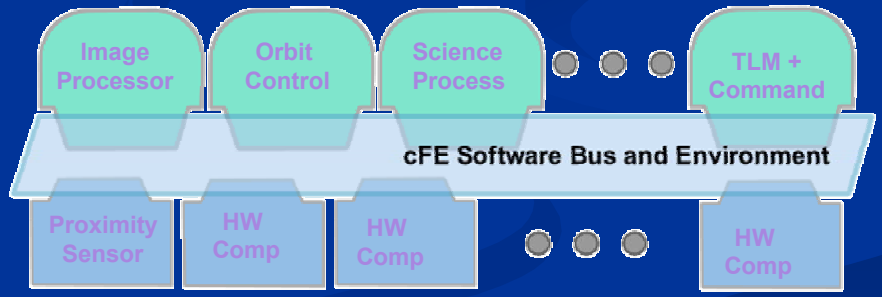
Reusable Components

- Common FSW functionality has been abstracted into a library of **reusable components and services**.
- **Tested, Certified, Documented**
- **A system is built from:**
  - Reusable components
  - Core services
  - Custom mission specific components
  - Adapted legacy components
  - Associated HW



Benefits:

- Reuse of tested, certified components supplies savings in each phase of the software development cycle:
- Reduces risk
- **Teams focus on the custom aspects** of their project and don't "reinvent the Flight Software wheel."



Packaging the Concepts

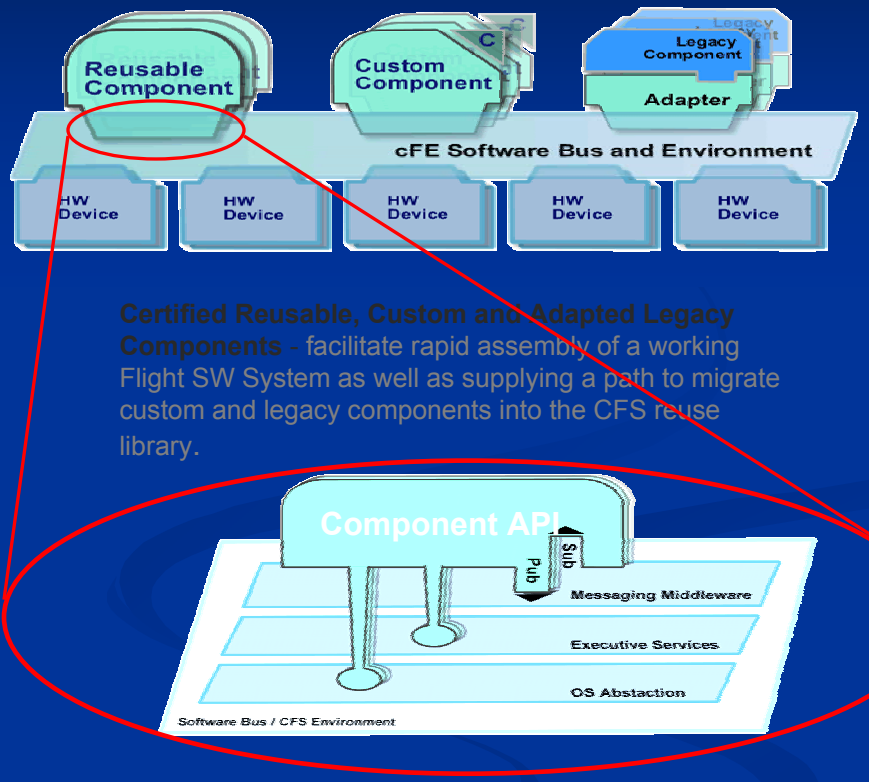
CFS Component Library

Flight Software Reuse Libraries

- Select pre-validated FSW components from FSW reuse libraries
- Validated common SW components include Requirements, Test Plan, Documentation

Mission-Unique Components

- Mission-unique FSW Components can be adapted for use
- Science applications developed on Scientist's desktop can plug into flight systems without change when developed with the API standards

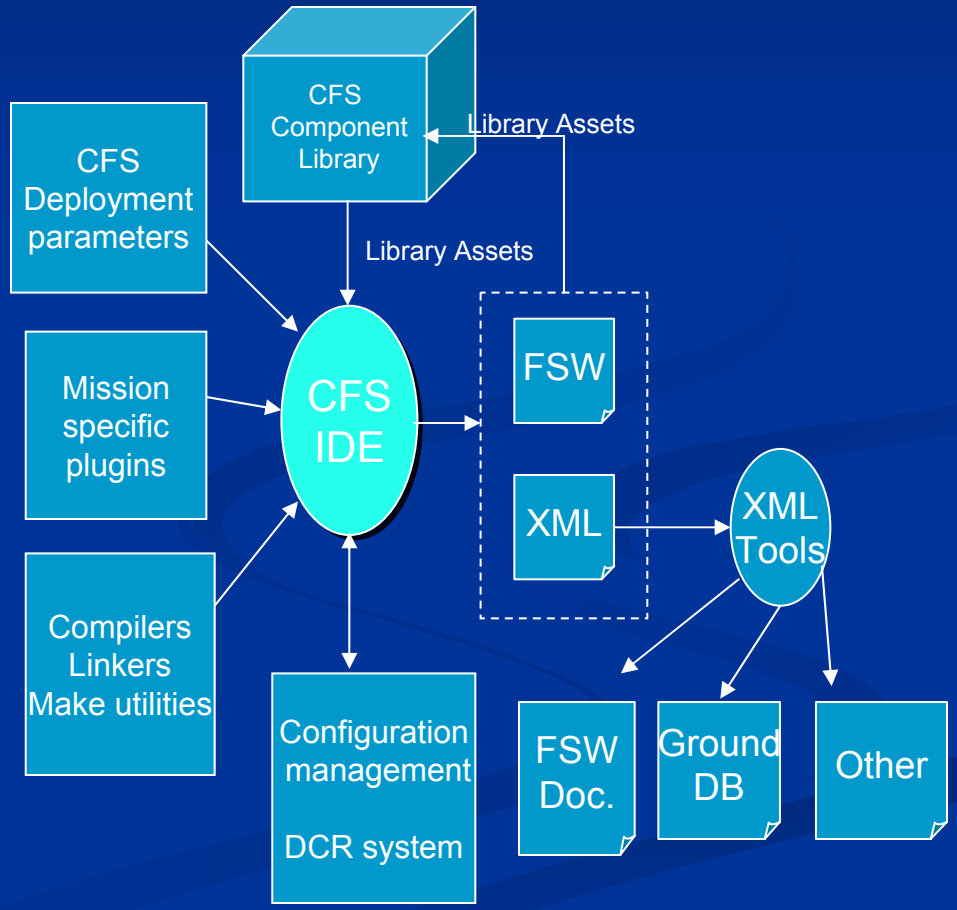


Certified Reusable, Custom and Adapted Legacy Components - facilitate rapid assembly of a working Flight SW System as well as supplying a path to migrate custom and legacy components into the CFS reuse library.

Component Standard API and Data Communication : components are loosely-coupled with a standard API and standard publish/subscribe intercommunications protocol and data formats. These features enable plug & play connectivity and dynamic integration / reconfiguration.

Integrated Development Environment (IDE)

- The CFS IDE is a growing set of integrated tools that support the management, development and configuration of a CFS deployment
- Tools are hosted on the open source Eclipse platform being adopted by many tool vendors, including Wind Rivers' VxWorks
- IDE supports rapid deployment using pick and click graphical user interface for system configuration



Status

- **Funding**
  - Received limited Mission funding
  - GPM, HRV, LRO = Customer investments; Some R&TD and GMSEC investments

- **2004 multi-CPU/Box prototype demonstrated**
  - Core, generic FSW services and Software components
  - **Dynamic application load and startup**
  - **Dynamic message bus reconfiguration for "Box faults"**
- Version 2 cFE, delivering to LRO July 15 2005
- VxWorks 6.0 integration to start July 17 2005
- IDE development started